

# Greedy Biomarker Discovery in the Genome with Applications to Antimicrobial Resistance

Alexandre Drouin<sup>1</sup>

Sébastien Giguère<sup>3</sup>

Maxime Deraspe<sup>2</sup>

François Laviolette<sup>1</sup>

Mario Marchand<sup>1</sup>

Jacques Corbeil<sup>2</sup>

ALEXANDRE.DROUIN.8@ULAVAL.CA

GIGUERE.SEBASTIEN@GMAIL.COM

MAXIME.DERASPE.1@ULAVAL.CA

FRANCOIS.LAVIOLETTE@IFT.ULAVAL.CA

MARIO.MARCHAND@IFT.ULAVAL.CA

JACQUES.CORBEIL@GENOME.ULAVAL.CA

<sup>1</sup> Department of Computer Science and Software Engineering, <sup>2</sup> Department of Molecular Medicine, Laval University, Quebec, Canada; <sup>3</sup> Institute for Research in Immunology and Cancer, University of Montreal, Montreal, Canada

## Abstract

The Set Covering Machine (SCM) is a greedy learning algorithm that produces sparse classifiers. We extend the SCM for datasets that contain a huge number of features. The whole genetic material of living organisms is an example of such a case, where the number of feature exceeds  $10^7$ . Three human pathogens were used to evaluate the performance of the SCM at predicting antimicrobial resistance. Our results show that the SCM compares favorably in terms of sparsity and accuracy against  $L_1$  and  $L_2$  regularized Support Vector Machines and CART decision trees. Moreover, the SCM was the only algorithm that could consider the full feature space. For all other algorithms, the latter had to be filtered as a preprocessing step.

## 1. Introduction

Genomics is a discipline of biology that focuses on analysing the entire genetic material of individuals, which is called the genome. Recent advances in next-generation sequencing (NGS) have led to a tremendous increase in the affordability of whole genome sequencing (van Dijk et al., 2014). The reduced cost and increased throughput of NGS have motivated its use for case-control studies, where groups of individuals are compared based on their genomes (Hall et al., 2013; van Dijk et al., 2014). Such studies can serve to determine the genomic variations that are biomarkers (i.e.: measurable characteristics) of a given

biological state (phenotype). Identifying such biomarkers has important implications in the clinical setting, where they can serve as the basis for diagnostic tests. Moreover, they can guide the development of new personalised therapies or drug treatments, by providing insight on the biological processes that are responsible for a phenotype.

It is common to represent a genome by a set of single nucleotide polymorphisms (SNP) (Brookes, 1999). A SNP exists at a single base pair location in the genome when a variation occurs within a population. They are obtained by aligning multiple genomes, a computationally expensive task that can be affected by gene deletions, duplications, inversions, or translocations (Leimeister et al., 2014). To address these limitations, we favor an approach, inspired by the “bag-of-words” representation, that is heavily used in the domain of text classification and string kernels. It consists in representing each genome by all its constituent  $k$ -mers, i.e. all the substrings of length  $k$  that are contained in the genome.

In the context of biomarker discovery, one is interested in finding the smallest subset of genomic features that allows to accurately predict the phenotype. Including superfluous features in this subset, would lead to the development of unnecessarily complicated diagnostic tests, generating additional costs. This is a challenging machine learning problem on many aspects. First, only a small fraction of the  $k$ -mer features are likely to be associated with the phenotype. Second, some  $k$ -mers are naturally highly correlated, as they belong to the same gene or gene family. Third, for genomes, the number of learning examples is often much smaller than the total number of possible  $k$ -mers. Therefore, one must use a method that favors sparsity and that is able to retrieve important features from such an extremely high dimensional feature space, while at the same time avoiding overfitting.

In this paper, we propose a method for learning sparse and interpretable models from whole genomes for predicting discrete phenotypes. Our approach relies on the Set Covering Machine (Marchand & Shawe-Taylor, 2003), a greedy learning algorithm that produces highly sparse models and that achieved state-of-the-art accuracy for many learning tasks, such as learning from DNA microarray data (Shah et al., 2012). The obtained models are short conjunctions or disjunctions of boolean-valued rules, which can explicitly highlight the importance of specific DNA sequences.

The next section presents the Set Covering Machine algorithm together with some improvements. Then, we explain how the SCM can be used to learn from genomes. Finally, the algorithm is used to predict the antimicrobial resistance of three common human pathogens for a panel of antibiotics. The results are then compared to the ones of  $L_1$  and  $L_2$  regularized Support Vector Machines (Cortes & Vapnik, 1995) and CART decision trees (Breiman et al., 1984) based on risk and sparsity.

## 2. Methods

### 2.1. The Set Covering Machine

In the supervised machine learning setting, we assume that data are available as a set  $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \sim D^m$ , where  $\mathbf{x}_i \in \mathcal{X}$  is a training example,  $y_i \in \mathcal{Y}$  its associated label and  $D$  is an unknown data generating distribution. We consider binary classification problems where  $\mathcal{Y} = \{0, 1\}$ . The goal of a learning algorithm is to produce a predictor  $h : \mathcal{X} \rightarrow \mathcal{Y}$  that minimizes the expected risk, which is given by:

$$E_{(\mathbf{x}, y) \sim D} I[h(\mathbf{x}) \neq y], \quad (1)$$

with  $I[True] = 1$  and 0 otherwise.

The Set Covering Machine (SCM) (Marchand & Shawe-Taylor, 2003), is a learning algorithm that produces predictors that are conjunctions or disjunctions of boolean-valued rules  $r : \mathcal{X} \rightarrow \{0, 1\}$ . Given a set of rules  $\mathcal{R}$ , the SCM attempts to find a predictor that minimizes the empirical risk  $R_{\mathcal{S}} \stackrel{\text{def}}{=} \sum_{i=1}^m I[h(\mathbf{x}_i) \neq y_i]/m$ , while using the smallest subset of  $\mathcal{R}$ . This problem is reducible to the minimum set cover problem, which is known to be *NP*-hard (Haussler, 1988; Marchand & Shawe-Taylor, 2003). To overcome this issue, the SCM uses a greedy optimisation algorithm inspired by the algorithm of Chvátal (1979), which yields an approximate solution with a worst case guarantee. In addition, Germain et al. (2012) used combinatorial optimisation to show that the solution found using the greedy heuristic is very close to optimality in most cases.

Algorithm 1 presents the SCM algorithm for the case where the returned predictor is a conjunction of boolean-valued rules. For the sake of conciseness, we only present the con-

junction case. The disjunction case can be obtained from the previous one by using  $\mathcal{S}' = \{(\mathbf{x}_i, \neg y_i) : (\mathbf{x}_i, y_i) \in \mathcal{S}\}$  as the set of training examples and taking the complement of the returned predictor  $h$ . This follows from the De Morgan law:  $\neg(\bigwedge_{r^* \in \mathcal{R}^*} r^*(\mathbf{x})) = \bigvee_{r^* \in \mathcal{R}^*} \neg r^*(\mathbf{x})$ .

The SCM starts with an empty conjunction and extends it in a greedy manner by iteratively selecting the rule maximizing a utility function. The latter is designed to favor conjunctions that correctly classify most of the training examples, while taking into account some constraints imposed by a greedy approach. Indeed, since the algorithm is greedy, once a rule is added to the conjunction, it cannot be removed. Observe that, any rule in the conjunction that assigns the negative class to an example forces the result of the conjunction itself to be negative. Therefore, there are two types of errors to consider: making an error on a negative example, which can be recovered, and making an error on a positive example, which cannot be recovered. For this reason, Marchand & Shawe-Taylor (2003) propose to score each rule using the following utility function:

$$U \stackrel{\text{def}}{=} |\mathcal{A}| - p \cdot |\mathcal{B}|, \quad (2)$$

where  $|\mathcal{A}|$  is the number of negative examples that it correctly classifies (i.e., covered by that rule) and  $|\mathcal{B}|$  is the number of positive examples on which it errs. A hyperparameter  $p$ , that is usually selected by cross-validation, allows to fix the correct trade-off between these two types of errors.

This process is repeated until a stopping criterion is reached. However, at each iteration the examples that are classified as negative by the selected rule are discarded for further computations of the utility function. This is justified by the observation above, that is, for those examples, the result of the conjunction is necessarily negative. This ensures that redundant rules are not added to the conjunction, effectively favoring sparse models.

There are 3 stopping criteria. The first stopping criterion is reached when all the negative examples have been covered by the rules of the conjunction. In this case, there is no need to continue extending the conjunction, as it is consistent with all the negative examples and adding more rules can only lead to more errors on the remaining positive examples. The second stopping criterion is reached when the number of rules in the conjunction reaches the limit  $s$ . This limit is a hyperparameter that induces regularization by early-stopping. Finally, the third stopping criterion is reached when the rule of maximal utility is consistent with no negative examples and errs on no positive examples. In this case, the algorithm is in a state of equilibrium, since no examples are removed at the end of the iteration.

Note that the version of the SCM presented here differs in two points from the one of Marchand & Shawe-Taylor

**Algorithm 1** Set Covering Machine (Conjunction)

---

**Input:**  $\mathcal{S}$ : A set of  $m$  training examples,  $\mathcal{R}$ : A set of boolean-valued rules,  $p$ : The trade-off parameter,  $s$ : The early stopping parameter

$\mathcal{R}^* \leftarrow \emptyset$

$\mathcal{P} \leftarrow$  the set of examples in  $\mathcal{S}$  with label 1

$\mathcal{N} \leftarrow$  the set of examples in  $\mathcal{S}$  with label 0

$stop \leftarrow False$

**while**  $\mathcal{N} \neq \emptyset$  **and**  $|\mathcal{R}^*| < s$  **and**  $\neg stop$  **do**

    ▷ Compute the utility function for each rule

$\forall i \in \{1, \dots, |\mathcal{R}|\},$

$\mathcal{A}_i \leftarrow$  the subset of  $\mathcal{N}$  correctly classified by  $r_i$

$\mathcal{B}_i \leftarrow$  the subset of  $\mathcal{P}$  misclassified by  $r_i$

$U_i \leftarrow |\mathcal{A}_i| - p \cdot |\mathcal{B}_i|$

    ▷ Select the best rule

$U^* \leftarrow \max_{i \in \{1, \dots, |\mathcal{R}|\}} U_i$

$\mathcal{C} \leftarrow \{i \in \{1, \dots, |\mathcal{R}|\} \mid U_i = U^*\}$

$i^* \leftarrow \operatorname{argmin}_{i \in \mathcal{C}} \sum_{j=1}^m I[r_i(\mathbf{x}_j) \neq y_j] / m$

    ▷ Add the best rule to the conjunction

**if**  $|\mathcal{A}_{i^*}| > 0$  **or**  $|\mathcal{B}_{i^*}| > 0$  **then**

$\mathcal{R}^* \leftarrow \mathcal{R}^* \cup \{r_{i^*}\}$

$\mathcal{N} \leftarrow \mathcal{N} - \mathcal{A}_{i^*}$

$\mathcal{P} \leftarrow \mathcal{P} - \mathcal{B}_{i^*}$

**else**

$stop \leftarrow True$

**endif**

**end while**

**return**  $h$ , where  $h(\mathbf{x}) = \bigwedge_{r^* \in \mathcal{R}^*} r^*(\mathbf{x})$

---

(2003). First, when more than one rule have the maximal utility, it selects the rule with the smallest empirical risk. This simple strategy is important for genomic datasets where the number of features is much greater than the number of examples. It becomes particularly important after a few iterations, as fewer examples contribute to the utility function and a lot of rules may have the same utility. In this situation, it is reasonable to assume that the rule that has the best performance on all the examples of the training set, is more likely to contribute to the best generalization performance. Second, the algorithm is stopped when it reaches the state of equilibrium mentioned above. This prevents from adding useless rules and reduces the training time.

The worst-case running time complexity of Algorithm 1 is  $O(|\mathcal{R}| \cdot |\mathcal{S}| \cdot s)$ . It thus scales linearly in the number of rules and the number of training examples.

## 2.2. Applying the Set Covering Machine to Genomes

We represent each genome by the presence or absence of every possible  $k$ -mer. Let  $\mathcal{K}$  be the set of all, possibly overlapping,  $k$ -mers present in at least one genome of

the training set. We can safely omit  $k$ -mers that are not in  $\mathcal{K}$  as they could not serve to discriminate genomes of the training set. For each genome  $\mathbf{x}$ , we define a vector  $\phi(\mathbf{x}) \in \{0, 1\}^{|\mathcal{K}|}$ , such that  $\phi_i(\mathbf{x}) = 1$  if the  $k$ -mer  $k_i \in \mathcal{K}$  is in  $\mathbf{x}$  and 0 otherwise. We then define a new training set  $\mathcal{S}' = \{(\phi(\mathbf{x}_i), y_i) : (\mathbf{x}_i, y_i) \in \mathcal{S}\}$ .

The set of boolean-valued rules that we consider is composed of 2 types of rules: presence rules and absence rules, which rely on the  $\phi(\mathbf{x})$  vectors to determine their outcome. For each  $k$ -mer  $k_i \in \mathcal{K}$ , we define a presence rule as  $p_{k_i}(\phi(\mathbf{x})) \stackrel{\text{def}}{=} I[\phi_i(\mathbf{x}) = 1]$  and an absence rule as  $a_{k_i}(\phi(\mathbf{x})) \stackrel{\text{def}}{=} I[\phi_i(\mathbf{x}) = 0]$ . The rules for each  $k$ -mer in  $\mathcal{K}$  are then combined to form the set  $\mathcal{R}$ .

The SCM (Algorithm 1), can then be applied with  $\mathcal{S}'$  as the training set and  $\mathcal{R}$  as the set of boolean-valued rules. This yields a predictor which explicitly highlights the importance of a small set of  $k$ -mers for predicting the phenotype. In addition, this predictor has a form which is simple to interpret, since its predictions are the outcome of a simple logical operation.

## 3. Results and Discussion

We applied the SCM and our proposed data representation to a real-world biomarker discovery problem, which consists in predicting the antimicrobial resistance bacteria based on their genomes. Antimicrobial resistance is a growing public health concern, as many multi-drug-resistant strains are starting to emerge. This compromises our ability to treat common infections, which results in an increasing number of deaths and health care costs (World Health Organization, 2014). An accurate predictor of antimicrobial resistance, could allow faster profiling of drug-resistant strains.

We present results for three human pathogens: *Clostridium difficile*, *Pseudomonas aeruginosa* (Kos et al., 2015) and *Streptococcus pneumoniae* (Croucher et al., 2013). For each of the latter, 285 to 556 bacterial isolates were collected from patients across the world. The genome of each isolate was sequenced and their susceptibility was measured against a panel of antibiotics. We considered each (pathogen, antibiotic) combination individually, yielding 12 datasets in which the number of  $k$ -mers ( $|\mathcal{K}|$ ) ranges from 10,542,251 to 132,487,288. Note that we consider  $k$ -mers of length 31, as this value is often used for bacterial genome assembly.

We empirically compared the risk and sparsity of models obtained using the SCM,  $L1$  and  $L2$  regularized SVMs (Cortes & Vapnik, 1995) and the CART decision tree algorithm (Breiman et al., 1984). We used the SVM implementation from LIBLINEAR (Fan et al., 2008) and the CART implementation from Scikit-learn (Pedregosa

Table 1. Results for the Set Covering Machine (SCM), the CART algorithm (CART),  $L_1/L_2$  regularized Support Vector Machines (L1SVM, L2SVM) and the baseline (Dummy). The prefix  $\chi^2$  indicates that a univariate filter was applied prior to learning. For each dataset, the values are the average risk and number of  $k$ -mers in the model (in parenthesis) for the 5 folds. The best risks are in bold.

DATASET	SCM	$\chi^2$ + SCM	$\chi^2$ + CART	$\chi^2$ + L1SVM	$\chi^2$ + L2SVM	DUMMY
<b>C. DIFFICILE</b>						
AZITHROMYCIN	<b>0.015</b> (3.2)	0.024 (4.8)	0.035 (6.6)	0.020 (494.6)	0.035 (2451870.2)	0.461
CEFTRIAXONE	<b>0.070</b> (2.0)	0.130 (5.6)	0.112 (7.2)	0.091 (277.8)	0.091 (2332313.0)	0.305
CLARITHROMYCIN	<b>0.015</b> (3.0)	0.019 (4.6)	0.026 (7.6)	0.022 (522.6)	0.041 (2426505.8)	0.461
CLINDAMYCIN	0.025 (2.0)	0.025 (2.4)	0.008 (2.4)	<b>0.006</b> (702.2)	0.03 (2405735.4)	0.140
MOXIFLOXACIN	<b>0.019</b> (1.0)	0.030 (1.8)	<b>0.019</b> (1.0)	0.022 (173.6)	0.048 (2432399.0)	0.407
<b>P. AERUGINOSA</b>						
AMIKACIN	<b>0.181</b> (6.0)	0.208 (9.8)	0.211 (18.8)	0.222 (687.8)	0.186 (164778.2)	0.230
DORIPENEM	0.234 (1.4)	0.237 (1.6)	0.242 (25.4)	<b>0.220</b> (44.8)	0.237 (16614.2)	0.377
MEROPENEM	0.280 (1.8)	0.272 (1.8)	0.283 (9.2)	<b>0.253</b> (233.6)	0.256 (3475.6)	0.416
LEVOFLOXACIN	0.067 (1.4)	<b>0.058</b> (1.8)	0.067 (1.0)	0.081 (180.4)	0.137 (173177.4)	0.472
<b>S. PNEUMONIAE</b>						
BENZYL PENICILLIN	<b>0.012</b> (1.0)	<b>0.012</b> (1.2)	0.012 (1.8)	0.019 (295.8)	0.017 (550134.8)	0.076
ERYTHROMYCIN	<b>0.031</b> (2.0)	0.047 (5.6)	0.045 (4.4)	0.034 (299.4)	0.041 (476701.6)	0.142
TETRACYCLIN	<b>0.025</b> (1.2)	<b>0.025</b> (2.2)	0.028 (1.0)	<b>0.025</b> (479.8)	<b>0.025</b> (516480.4)	0.111
AVERAGE	<b>0.081</b> (2.2)	0.091 (3.6)	0.091 (7.2)	0.085 (366.0)	0.095 (1162515.5)	0.300

et al., 2011).

The great number of features that we consider poses computational challenges in terms of runtime and memory usage. The simplicity of the SCM and its low computational complexity enabled us to implement the algorithm in a way that made it possible to learn from the entire feature space. However, for SVM and CART, dimensionality reduction was necessary. For these algorithms, we filtered the features using a univariate filter (Guyon & Elisseeff, 2003), with the  $\chi^2$  test as the measure of significance and the method of Benjamini & Yekutieli (2001) for multiple testing correction.

Table 1 presents the 5-fold nested cross-validation risk and the average number of  $k$ -mers in the model for each dataset and learning algorithm. For each fold, the hyperparameters were selected using standard cross-validation on the remaining 4 folds. This table also includes a comparison to a baseline (Dummy), that predicts the majority class in the training set.

Observe that the SCM tends to learn models that are much sparser than the ones of SVMs. This is an interesting result, as both the SCM and the L1SVM algorithms attempt to obtain sparse solutions. This suggests that the greedy heuristic of the SCM is much more efficient at minimizing the  $L_0$  norm than the L1SVM. Moreover, although the difference in sparsity is less striking, the SCM tends to learn sparser models than CART.

In addition, note that all the algorithms clearly outperform the dummy predictor, which means that some information on antimicrobial resistance is contained in the genomes. It can also be observed that, for 8 of the 12 datasets, the risks

of the SCM predictors are smaller or equal to the ones of the other algorithms. This suggests that the extreme sparsity of the former does not undermine their generalization performance.

Finally, we compared the SCM to a variant which uses a univariate filter as a preprocessing step ( $\chi^2$ + SCM). On some datasets, using such a filter leads to an increased risk and denser models. Being able to consider the entire feature space without filtering is thus an interesting property of the SCM algorithm.

## 4. Conclusion

In this work, we have confronted the Set Covering Machine to the challenging problem of learning from extremely high dimensional feature spaces and obtaining sparse models. The analysis was conducted in the context of biomarker discovery, which is a problem of high importance. We showed that, as opposed to other learning algorithms, the Set Covering Machine can learn from entire genome sequences without requiring prior feature selection. Our results for predicting antimicrobial resistance suggest that the greedy heuristic of the SCM produces sparser models than a Support Vector Machine with a  $L_1$  regularizer, while having similar and often better generalization performance. To the best of our knowledge, this is the first time that Set Covering Machines are used on datasets of such high dimensionality. The fact that the obtained models are sparse and generalize well, opens the door to new applications in other fields where datasets of high dimensionality are common, such as genome-wide association studies (GWAS) and natural language processing.

## Acknowledgments

We thank Dr Veronica Kos, Dr Humphrey Gardner and their colleagues from AstraZeneca for providing the *Pseudomonas aeruginosa* data. We also thank Dr Vivian Loo and Dr Anne-Marie Bourgault for sharing the *Clostridium difficile* data. Computations were performed on the Colosse supercomputer at Université Laval (resource allocation project: nne-790-ae), under the auspices of Calcul Québec and Compute Canada. AD is recipient of an Alexander Graham Bell Canada Graduate Scholarship Doctoral Award from the National Sciences and Engineering Research Council of Canada (NSERC). This work was supported in part by the Fonds de recherche du Québec - Nature et technologies (FL, MM & JC; 2013-PR-166708), the NSERC Discovery Grants (FL; 262067, MM; 122405) and an award to Michael Tyers from the Ministère de l'enseignement supérieur, de la recherche, de la science et de la technologie du Québec through Génome Québec (SG). JC acknowledges the Canada Research Chair in Medical Genomics.

## References

- Benjamini, Yoav and Yekutieli, Daniel. The control of the false discovery rate in multiple testing under dependency. *Annals of statistics*, pp. 1165–1188, 2001.
- Breiman, Leo, Friedman, Jerome, Stone, Charles J, and Olshen, Richard A. *Classification and regression trees*. CRC press, 1984.
- Brookes, Anthony J. The essence of snps. *Gene*, 234(2): 177–186, 1999.
- Chvátal, V. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, 4(3):233–235, August 1979.
- Cortes, Corinna and Vapnik, Vladimir. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- Croucher, Nicholas J, Finkelstein, Jonathan A, Pelton, Stephen I, Mitchell, Patrick K, Lee, Grace M, Parkhill, Julian, Bentley, Stephen D, Hanage, William P, and Lipsitch, Marc. Population genomics of post-vaccine changes in pneumococcal epidemiology. *Nature genetics*, 45(6):656–663, May 2013.
- Fan, Rong-En, Chang, Kai-Wei, Hsieh, Cho-Jui, Wang, Xiang-Rui, and Lin, Chih-Jen. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874, June 2008.
- Germain, Pascal, Giguere, Sébastien, Roy, Jean-François, Zirakiza, Brice, Laviolette, François, and Quimper, Claude-Guy. A pseudo-boolean set covering machine. In *Principles and Practice of Constraint Programming*, pp. 916–924. Springer, 2012.
- Guyon, Isabelle and Elisseeff, André. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, March 2003.
- Hall, Barry G, Cardenas, Heliodoro, and Barlow, Miriam. Using complete genome comparisons to identify sequences whose presence accurately predicts clinically important phenotypes. *PloS one*, 8(7):e68901, 2013.
- Haussler, D. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial intelligence*, 36(2):177–221, 1988.
- Kos, Veronica N, Deraspe, Maxime, McLaughlin, Robert E, Whiteaker, James D, Roy, Paul H, Alm, Richard A, Corbeil, Jacques, and Gardner, Humphrey. The resistome of *Pseudomonas aeruginosa* in relationship to phenotypic susceptibility. *Antimicrobial Agents and Chemotherapy*, 59(1):427–436, January 2015.
- Leimeister, Chris-Andre, Boden, Marcus, Horwege, Sebastian, Lindner, Sebastian, and Morgenstern, Burkhard. Fast alignment-free sequence comparison using spaced-word frequencies. *Bioinformatics*, pp. btu177, 2014.
- Marchand, Mario and Shawe-Taylor, John. The set covering machine. *The Journal of Machine Learning Research*, 3:723–746, 2003.
- Pedregosa, Fabian, Varoquaux, Gaël, Gramfort, Alexandre, Michel, Vincent, Thirion, Bertrand, Grisel, Olivier, Blondel, Mathieu, Prettenhofer, Peter, Weiss, Ron, Dubourg, Vincent, Vanderplas, Jake, Passos, Alexandre, Cournapeau, David, Brucher, Matthieu, Perrot, Matthieu, and Duchesnay, Édouard. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, February 2011.
- Shah, Mohak, Marchand, Mario, and Corbeil, Jacques. Feature selection with conjunctions of decision stumps and learning from microarray data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(1):174–186, 2012.
- van Dijk, Erwin L, Auger, Hélène, Jaszczyszyn, Yan, and Thermes, Claude. Ten years of next-generation sequencing technology. *Trends in Genetics*, 30(9):418–426, 2014.
- World Health Organization. *Antimicrobial resistance: global report on surveillance*. World Health Organization, 2014.